Freedom and forking in open source software: the MariaDB story

LINUS NYMAN
Hanken School of Economics
linus.nyman@hanken.fi

Abstract

While significant factors that affect the open source community's interest to participate in a development project have been studied, there has been little focus on the motivating factors that can cause a contributor to become a competitor by utilizing the right to fork a program i.e., to copy an existing program's code base and use it to begin a separate development.

The right to copy an existing program's code base and use it to begin a separate development is guaranteed by all open source licenses. However, this right to fork a program is rarely exercised. Indeed, there is strong social pressure against code forking stemming from the negative side effects of code forking, such as conflict and duplicated efforts among developers.

This paper details the events that led Widenius, the founder of the MySQL project, to decide to fork MariaDB from MySQL. Our findings confirm the previously held notion that there is a high threshold for starting a competing fork. While the few studies that exist of competitive forks find the reasons to be due to disagreement among developers, in the case of MariaDB the fork was caused by Widenius' concerns regarding the uncertainty of the future freedom and openness of the MySQL codebase.

This article makes three contributions. Firstly, it further validates the existing notion that there is a strong threshold to starting a competing fork. Secondly, it offers an in-depth analysis of the events and motivations behind the birth of a fork. Thirdly, it contributes to theory by introducing the freedom factor hypothesis: limiting either developers' freedoms to contribute to a project or the freedom inherent in a project's license increases the likelihood of a fork.

Keywords

Code forking, open source software development, open source business models

Introduction

Before software became a viable market in itself, there was not "open source" or "proprietary" software; there was just software. The computer hardware industry saw software as a tool through which one could access their proprietary hardware. Therefore, they freely provided software to be distributed with their hardware. At this time, much of what users needed was developed by the users themselves sharing code and ideas among one another. Far from being frowned upon, this practice was actually encouraged by most vendors, as it aided the sale of hardware (Levy, 2010). Out of this group of people who developed and shared software evolved the "hackers". Many of these hackers were MIT computer enthusiasts with their own subculture, values and ethic. Among the central tenets in the hacker ethic were openness, the sharing of knowledge that others can benefit from, and the freedom of information (ibid.). A further element of the hacker attitude was that "no problem should ever have to be solved twice", with closed source licensing (i.e. proprietary software) commonly

seen as erecting "artificial technical, legal, or institutional barriers […] that prevent a good solution from being re-used and *force* people to re-invent wheels" (Raymond, 1999*a*; emphasis in original).

Therefore, the open source development model has its roots in the early 'hacker' culture. As computers became more affordable, and thus more commonplace, the need for software grew. In 1969 IBM unbundled their software and hardware (IBM Archives), considered a pivotal event in the birth and growth of the commercial software industry (e.g. Grad, 2002), and with it the growth of the proprietary license. Richard Stallman, who had been a part of the hacker community at MIT, saw the rise of proprietary code around him and wanted to "create a new software-sharing community" (Stallman, 1999). In 1984, in what can arguably be called the beginning of the free (and open source) software movement, Stallman began the GNU project, the goal of which was to create a free, and freely sharable, operating system (ibid.)[1]. In essence, the free and open source movement began not so much to instigate a change in software licensing, but to ensure that software development could continue on as it had before: with free access to code and the right to modify and share programs according to one's own needs.

Over time, corporations began producing, acquiring and distributing open source software, creating a dynamic in which the community desire for freedom and access to the code must coexist with corporate needs. While the direct profitability of a specific open source project may not be the goal of a corporation (e.g. Lerner and Tirole, 2002; West, 2003), corporate participation in the open source market necessitates a significant balance between the needs of the community and those of the corporation.

Types of OS projects, business models, and architectures of participation

Among extant categories of open source projects, a common dividing factor is that of community versus corporation as the owner and driving force behind the project (e.g. West and O'Mahoney, 2005; Markus, 2007; O'Mahoney, 2007). West and O'Mahoney (2008) use the classification terms "autonomous" versus "sponsored" projects. An *autonomous* project is a community-developed project, in which governance and control are shared widely among the community. In some cases, a non-profit foundation is created to support autonomous projects and delineate ownership. In a *sponsored* open source project, the community's short or long-term activities are controlled by one or more corporate entities (ibid.). Figure 1 shows a development model for a sponsored (or "corporate") open source project. In this model, the primary driving force is the sponsor company. However, the project also receives ancillary support from the open source community and corporate community, normally made up of companies that use the software or for whom its development is otherwise significant. Members of the corporate community commonly contribute either money or developer time to the project.

[1] For more information, see http://www.gnu.org/gnu/thegnuproject.html

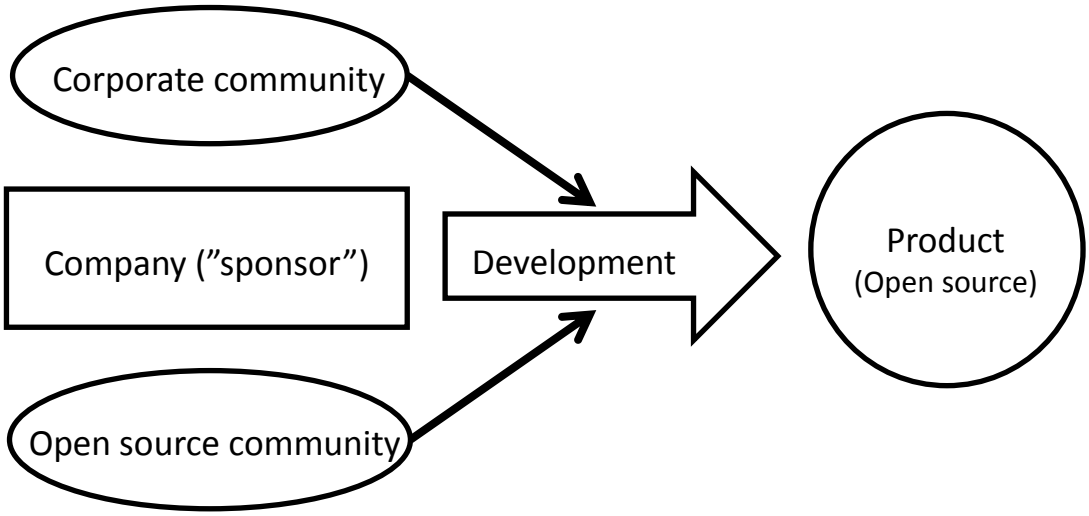

Figure 1: Example of a corporate (“sponsored”) open source project development

Sponsored and community projects differ both in their developer communities and in their architectures of participation (West and O’Mahoney, 2008). One key difference is that sponsored projects need to manage the tension between sponsor and participant goals, essentially community desires and profit. An array of business models have emerged to meet the financial needs of sponsored projects. A central difference between these business models is whether or not the entire product is offered under an open source license. Among those that do offer their entire product as open source, perhaps the most common means of generating income is through the sale of support and other services. Here, the software can be downloaded free of charge and income is generated through providing support and services for the software. Another monetization strategy is to license part of a program as open source, with commercial (i.e. closed source) extensions sold to complement the open source element. This approach is commonly called “open core”, as the “core” of the program is open source. A third approach is dual licensing, in which the program is made available under an open source license, but the company also sells a closed source version of the same program. This is typically of interest mainly for companies that need to embed the company's software within their own proprietary software.

Licensing concerns are known to be a significant factor in corporate software adoption decisions (Daffara, 2011); however, the question of how business models and licensing affect community participation has seen little direct study. The handling of different licenses and using them “in a fruitful manner” is a substantial challenge in the shaping of a corporation's relationship to its open source community (Dahlander and Magnusson, 2005). In fact, “being specific about licensing practices is a prerequisite for firms to be trusted in the open source community” (Dahlander and Magnusson, 2008). Bonaccorsi and Rossi (2006) note that there is an “’implicit promise’ based on the non-written rules of the Free Software community”; if a company breaks this promise, developers are likely to stop cooperating or migrate to other projects. Similarly, Bacon (2009) notes that “the governance body should be tasked with the responsibility of always maintaining and defending the primary values of the community and standing up against any improper requests that may result from commercial sponsors”. While the hacker culture is not intrinsically against profitability, it has been documented that some members of the community object to profitability when project income is generated at the cost of openness. This situation is exemplified by the open core approach to

licensing. On one side of the debate are those that underline the importance of generating income to fund future development efforts, on the other are those that consider open core to be a "bait-and-switch […] offering the promise of open source but not delivering it" (e.g. Phipps, 2010). Indeed, Dahlander and Magnusson (2005: 490) note that "the developers and users of the Roxen web server went to other projects after the firm released its proprietary add-on". Furthermore, of the four companies they studied, those using an open core business model were less successful in attracting and maintaining a community than those projects that were entirely open source (ibid.). A final community concern worth noting is that of a corporation "hijacking" the code (e.g. Lerner and Tirole, 2002; Ciffolilli, 2004) by changing a project's source code from an open source to a closed source license.

Code forking; when participation becomes competition

As early as the late 1960s, the reuse of code was proposed as a means of building large, reliable software systems in a controlled, cost-effective way (Naur and Randell, 1969; Krueger, 1992). Today, all open source licenses guarantee the right to reuse a program's code (see the Open Source Initiative's open source definition at opensource.org/osd). Thus, it is perhaps not surprising that code reuse has become a common practice in open source software (Haefliger, von Krogh, and Spaeth, 2008). Furthermore, open source licenses do not restrict the amount of code that can be reused, meaning programmers can even reuse entire programs. Such a practice, taking an entire code base to use as the base for a new project, is commonly called forking the code, or "code forking". The ability to fork has a significant effect on the governance and sustainability of open source software (e.g. Nyman et al. 2011 & 2012; Nyman and Lindman, 2013).

The majority of forks[2] are benign, started for a variety of non-competitive reasons such as: to modify a program to better suit their needs; as a means of experimenting with new solutions or features with the intent of merging them back into the original project[3]; or because the original has been abandoned and is no longer being actively developed (Nyman & Mikkonen, 2011; G. Robles and J.M. González-Barahona, 2012). However, a fork may also occur due to disagreements among the developers, resulting in the splitting of the developer group and the forming of a separate, competing project (e.g. Raymond, 1999*b*; Fogel, 2009). This situation will be termed "competing forks" for the purposes of this paper. While even a situation in which the forks do compete can have benefits (e.g. Nyman et al. 2012), developers seek to avoid such forks as they are likely to result in a duplication of effort, a splitting and confusing of the community, as well as the potential demise of one (or both) of the projects. Therefore, while open source licenses guarantee developers the right to start a competing fork, social pressure discourages the exercising of that right (e.g. Meeker, 2008). Indeed, such cases are rare enough to be "remembered in the hacker folklore" (Raymond, 1999*b*).

Robles and González-Barahona (2012) studied 220 forks, noting that only 16 of the forks (7.3%) were due to differences among the developer team. Similarly, a study focused on the motivations developers stated for starting forks found only a few indications of differences of opinion[4] and no indications that disagreement among the developers influenced the decision to fork (Nyman and Mikkonen, 2011). Given the rarity of competing forks, it is unsurprising that there is little knowledge of the causes of these forks. Gamalielsson and Lundell (2012) studied the LibreOffice competing fork of OpenOffice, finding that a competing fork can be sustainable over time. However, beyond noting that it had occurred due to tensions in the original project, an in-depth analysis of the motivation behind the fork was not an area of focus. While the limited studies that exist support the

[2]As the term is used by OS developers; among academics the term fork is commonly more restrictive.
[3]Sometimes called "branches" rather than forks.
[4]With licensing being the most common motivating factors stated in such cases

notion that competitive forks are rare, we still know little about the exact nature of the disagreements that spawned them. Given that competitive forks are so actively avoided, cases in which they do happen are interesting extremes worthy of study.

Aim and Method

Given the scarcity of studies of the root causes of competitive forks, we examine one case in-depth to answer the question: why was the MariaDB fork started? Secondly, this article aims to determine why the fork was not instigated by earlier conflicts[5]. To answer the research questions, this paper details and analyzes a single-case study (Yin, 2009). The primary source of empirical data is based on interviews with Michael ("Monty") Widenius, the founder of both MySQL and MariaDB. Two in person interviews were conducted, averaging 45 minutes in length. Email and phone conversations were used for follow-up questions. The data set, consisting of transcriptions from the interviews, was complemented with archival research for triangulation (ibid.). The archival research consisted of Widenius' blog; press releases and other relevant information available on the MariaDB, MySQL, MariaDB Foundation, Monty program and Oracle websites, industry articles about MariaDB as well as interviews with Widenius reported in the following journals: ArsTechnica, Computer World, Forbes, H-Online, Info World, ITWire, Linux Journal, Techradar, and ZD Net. We also conducted searches for MariaDB, Widenius, Sun, and Oracle on community site Slashdot, and included all relevant articles. This author had interviewed Widenius once previously for a separate project. The transcripts from that interview were also included in the data used in the creation of this case study.

Most of the secondary data was gathered and examined prior to the personal interview. The first personal interview was used to clarify the motivations for the fork and fill in the gaps left by the secondary data. The second interview was used to ensure that the representation and interpretation of the case were accurate and complete. As the purpose of this paper was to describe why MariaDB was forked, I have chosen to focus on the viewpoints of Widenius, the person behind the fork, and have not asked other parties of their viewpoints on the matter.

Discussion

This article began with a discussion of a common means of development for sponsored open source projects, an approach also adopted by MySQL. It went on to discuss business models, licensing, architectures of participation, as well as the concept of code forking. We will now discuss in further detail the issues and events relevant to the research questions: why was the MariaDB fork started and are forks actively avoided. A list of significant events during the lead up to the birth of MariaDB is presented in Table 1.

Table 1: Significant events leading up to the MariaDB fork

2001	MySQL takes on outside financing. The shareholder agreement includes a section forbidding the changing of MySQL's license without the consent of the founders (Widenius and Axmark), thereby guarding against a switch to a proprietary or open core business model. The same clause is included in all subsequent financing rounds.
2008, January	**MySQL is acquired by Sun**. (Widenius becomes financially independent.)
2008, January	First **Maria engine** is released. Attempts are made on MySQL's part to stop release unless Widenius changes the name. Widenius replies "Try and stop me."

[5] Assuming there is a certain amount of conflict in any given organization, it raises the question *why now*, rather than as a result of any earlier conflict.

2008, April MySQL CEO announces MySQL will include commercial extensions. Widenius informs Sun about the announcement; Sun is against it.

2008, May MySQL CEO announces that MySQL is and will remain open.

2008, Spring/Summer **Developers** unhappy at Sun, say (Widenius notes) that they want to be moved to work under Widenius or they will leave.

2008, Fall **Widenius** leaves Sun, posts an open invitation for MySQL programmers to join his new company Monty Program. The majority ("All but one") of the core MySQL developers join him.

2009, April **Oracle and Sun** announce acquisition plans [6] [7]

2009-2010 **Widenius** petitions to have MySQL given to third party or to have its licensing changed (see Widenius' blog [8] for more detail). Attempts fail.

Research question 1: why was the MariaDB fork started?

After having faced many adverse events, the tipping point came when Widenius no longer felt confident in the future well-being and openness of the MySQL codebase. MySQL had been a strong competitor for Oracle, Widenius notes, in providing an open source alternative (with a dual licensing option) to Oracle's database management product. Widenius believed that Oracle would use the purchase of MySQL to generate income for Oracle in one of two ways: either by "killing" MySQL and thereby gaining customers and market share for their product, or by changing MySQL to an open core business model by combining a mixture of open and closed source parts. Both of these scenarios were unacceptable to Widenius, and the spectre of such actions provided the impetus for starting a competing fork.

Scenario 1: The "killing" of MySQL

With the acquisition of MySQL by Sun then later by Oracle, the companies were essentially paying for control over the project. With this control, the company was then responsible for deciding what contributed code to include in each distribution. They were therefore able to decide what to release when. Even if the open source community or corporate communities would like to participate in the development of an open source program, it is possible for the governing company to turn down this aid, for instance by not including contributed patches or bugfixes developed by the communities. Given that existing versions can be shared freely among interested users, an open source product cannot be killed quickly or immediately by pulling it off the market, as can happen with closed source products. However, open source products can be starved through not including improvements or updates into the codebase and thereby die a slow death. The result would be a program that ceases to evolve and meet the changing needs of its users. Software that does not evolve becomes less and less satisfactory to its users over time (Lehman, 1980), and sooner or later results in the user abandoning the program in favour of a more up-to-date program. In such a situation, the only means of keeping the program "alive" is the forking of the original to continue development on a competing version.

Scenario 2: Commercial extensions

The second possible outcome for the MySQL codebase was that Oracle would change MySQL's business model to an open core model by adding commercial extensions. Widenius is among those

[6] http://www.oracle.com/us/corporate/press/018363
[7] http://www.sec.gov/Archives/edgar/data/709519/000119312509126389/ddefm14a.htm
[8] http://monty-says.blogspot.fi/2009/12/help-saving-mysql.html

who are not favourable to the open core approach. In fact, during one interview he shared that, in his opinion, "open core is not open source". Even prior to its acquisition by Sun, MySQL's management team had been trying to implement an open core model, but Widenius had been able to block such attempts. If My SQL adopted an open core model, development of the codebase could continue, with community and corporate contributions still accepted for consideration, but not all of the resulting product would be open source.

Common to both scenarios discussed is that the solution to maintaining the openness and further development of the code was to fork the program. In a scenario in which the owners of the code restrict its development, a fork would be necessary to be able to continue developing the code. In a scenario in which the owners of the code switch to including proprietary extensions, a fork would be necessary to maintain a version in which all future additions to the code remain open source. While Widenius did not know at the time which of these would happen, or even if they would happen at all, he viewed the possibility as enough of a threat to the future openness of the code to warrant a fork. Hence, MySQL was forked and MariaDB born in order to ensure that the program would be continued as Widenius intended: constantly evolving and entirely open source.

Although Oracle would have known that a fork was a possibility, Widenius believed that Oracle simply considered it exceedingly unlikely that anyone would be willing to invest heavily enough into a fork to be able to make it successful. While forking the code itself is relatively easy, gaining development, marketing, consulting and services, etc. to support the project is a mammoth task, one requiring no small investment in time and money. Indeed, at the time of the interviews, Widenius had already invested several million Euros in the continued development of the codebase. Due to the sale of MySQL to Sun, Widenius was financially in a position that made it possible to invest money in developing the Maria-branch and the subsequent MariaDB fork. Furthermore, existing developers who already were experts in the code base were interested in working with him, a factor which enabled MariaDB to continue developing with little to no loss of time for the training of new talent. While a fork of the magnitude of MariaDB would technically be possible without money for development, in practice it would be a significantly more challenging endeavour to rely on volunteers working in their spare time[9].

Research question 2: why was the fork not instigated by earlier conflicts?

There were several conflicts during MySQL's existence before the actual fork, among them were the push for competing business models, internal disputes and disagreements regarding the quality of the product.

Competing business model interests. The original (and all subsequent agreements) MySQL shareholder agreement included a section that requires the consent of the founders Widenius and Axmark to change MySQL's license. Over the years, the MySQL management team made multiple attempts to change the MySQL business model to include commercial extensions. However, Widenius always rejected these requests given his strong support for the openness of the project. When MySQL was sold to Sun, the shareholders agreement was made void and Widenius no longer had the authority to stop such changes. Shortly after the acquisition by Sun, in the spring of 2008, MySQL management announced that it would include commercial extensions in the project. Widenius believed that Sun was unaware of this move, so therefore made no move to create an entirely open fork at that time. When Widenius confronted the managers at Sun regarding this change,

[9]As a point of reference, Widenius estimates that to learn the MariaDB or MySQL code base in its entirety would take 2-3 years when studied under the tutelage of someone already knowledgeable in it.

they assured him that they were unaware of the plans. Soon after, the announcement was retracted[10], with MySQL management noting that MySQL is and will always be free. During the codebase's time under MySQL it was kept from being changed to an open core model by the shareholders agreement's requirement of Widenius' consent; during it's time under Sun it was kept from being changed to open core by Sun management's view that MySQL should be open source.

Internal disputes. The Maria storage engine, out of which MariaDB evolved, was a topic of dispute, both regarding its name and the choice of the engine itself as a product. MySQL had used a storage engine called InnoDB. After Oracle, at the time one of MySQL's biggest competitors, purchased InnoDB, MySQL started looking for a replacement. They searched for a replacement over a year with no luck, so Widenius led a team of developers working part-time to develop a new storage engine. Meanwhile, the management team at MySQL believed that the storage engine Falcon, which MySQL had acquired, was the better choice to focus on, rather than waiting for Maria to be completed. In Widenius' opinion, the Maria engine's development received insufficient manpower and support from the MySQL project. Additionally, the naming of the Maria engine sparked further disagreement[11]. During his time at Sun, Widenius' primary obstacle was not direct conflict, but unresponsiveness. He notes that, while initially eager to "help make Sun a better open source company", those he contacted in the organization seemed to ignore his attempts to help. Widenius eventually felt that the best way to continue to develop the MySQL code base was to leave Sun. Even having left Sun, taking many key MySQL developers with him, the goal was not to fork the codebase. On the contrary, he considered leaving Sun to be the best way to focus on the continued improvement of MySQL through improving his experimental version with the goal of implementing the improvements in the original (by selling the improvements to Sun).

Quality of the product. Widenius felt Sun had released a MySQL version lacking in several key areas. While open about his discontent with the release, his goal was still an improved MySQL, not a fork.

Faced with each of these challenges Widenius never planned to fork the MySQL codebase. These findings further support to the notion that there is a high threshold for competitive forking, but raise the question of why the fork was finally started after avoiding it for so many years.

The freedom factor hypothesis

The absence of a fork during the earlier conflicts highlights the active avoidance of competitive forks ingrained within the open source community. Due to this competing fork-averse culture, the presence of conflicts alone are insufficient to account for the motivational drive behind at least some competing forks.

Earlier studies have discussed factors that enable and increase participation. It has been noted that managerial actions not supported by the developers can lead some developers to abandon a project, but there is a lack of knowledge regarding the factors that instigate a fork. The MariaDB case shows

[10]When asked to hypothesize about what he would have done had MySQL's management team managed to change MySQL to open core, Widenius notes that there was a group of developers in MySQL who had formed, ready to spit off into a separate entity to continue work on an open source version of MySQL. However, they would have attempted to work with, rather than against, MySQL, with the goal of achieving the best code possible. In other words, even then they would have attempted to avoid a fork. When asked if he would ever consider joining forces with Oracle to improve the code together, Widenius noted that he would do it "immediately" if the openness of the code were guaranteed.

[11]When it was ready for release, MySQL marketing told Widenius that it should be renamed after a bird, to fit with the Falcon engine, before he can release it. Widenius notes that his reply was "try and stop me", and he released it as Maria.

us a set of circumstances under which a contributor goes from contributor to competitor, rather than the more studied transition of contributor to non-contributor. This transition only occurred after Widenius felt there was a credible threat to the future openness of the project. In order to more fully understand the motivations behind this transition, we propose the freedom factor hypothesis: *limiting either developers' freedoms to contribute to a project, or the freedom inherent in the current license, increases the likelihood of a fork.*

The first part, the freedom to contribute, covers concern over the ability to contribute to the code. There is some support for this hypothesis in the "community forks" of projects, which are sometimes born to enable the community to contribute code more freely, without having to abide by the decisions of the sponsor regarding what is accepted into the program. Furthermore, in extreme cases where a project has been abandoned, it is not unusual that a fork of the program is started to breathe new life into the project (Nyman and Mikkonen, 2011; Robles and González-Barahona, 2012). The second part, the freedom of the license, concerns a change toward a more commercialized software license.

Conclusions

The open source movement was born from the hacker ethic of freedom and sharing. This philosophy is still present in the open source community, and can motivate developer disengagement or even a fork if these freedoms are seen to be hampered or threatened by moves to restrict the evolution or distribution of the program. Competitive forking, where a developer may go from collaborator to competitor, is considered to be the most extreme expression of this desire for freedom and sharing. While the right to fork is guaranteed by all open source licenses, our findings offer further proof of the commonly held notion that actors in the open source community go to great lengths to avoid starting competitive forks. Disagreements among the developers has previously been considered to be a leading cause of forks in open source projects. Conversely, analysis of the events leading up to the MariaDB fork shows that the impetus behind the fork was a perceived threat to the freedom to contribute to the code, its wellbeing, and the future openness of the code base.

This study shows that, in the world of open source software, ownership without trust is exceptionally fragile. Our findings suggest that managers/organizations must convey an unambiguous guarantee of the future openness of the code in order to achieve the greatest potential community contribution. Programmers must feel able to contribute code to a continuously developing program, which they feel confident will remain open source. If these criteria are not met, the likelihood of a fork increases, as such a situation will be unacceptable for at least some programmers. The results are explained through introducing the concept of the "freedom factor hypothesis": limiting either developers' freedoms to contribute to a project or the freedom inherent in the current license increases the likelihood of a fork. Further study is needed to validate this hypothesis.

Avenues for future study

Open core, business models & licensing. In sponsored open source projects, the hacker mentality and values co-exist with corporate interests and needs. This co-existence can work as long as the needs are not in significant conflict. We posit that the difference between open source and open source with proprietary extensions, or open code, is a significant one. Unfortunately, extant research does not draw a sufficiently clear line between open source and open core and is therefore of limited use in understanding the causal implications of this form of licensing. Further research into clarifying the open source community's views of open core would offer important insights into a topic that is of great significance to both generating income and the attracting and maintaining of contributors. In his

blog, Widenius notes that "You can't buy an open source project with money; the currency in open source is trust." There are open core projects that have significant contributor communities and those that do not. Is the difference one of trust? What is the source of that trust? A further significant future avenue of research here would be to find out how large a group is required to precipitate a competing fork.

The business models and project types of forks vs. parents. While Oracle can generate income through the dual-licensing of MySQL, MariaDB no longer owns the necessary code to do so. This creates a challenge in the financing of its development efforts. Future studies could examine the business models of forks vs. their parents to see how successfully forks manage to compete. Is it predominantly through a services model, or are there other approaches? Furthermore, do such forks commonly move from sponsored projects to community, or foundation-run projects, as in the case of MariaDB? Such future research could inform future managers on how to more effectively manage a project, as well as future developers considering a fork.

Motivations for switching. While anyone can start a fork, without developers, customers and community it will be short-lived. MariaDB has been gaining support among all these groups. Why are they switching over? Some cite concerns over the future openness of the MySQL codebase, others a desire to be closer to the open source spirit of things, while still others note the technical superiority of the MariaDB codebase. There is still much research to be done in this area to get a clear picture of both the motivating factors and their implications.

Categorization/taxonomy. Competitive forks are commonly thought to be caused by disagreements among the developers. A clearer understanding of what types of disagreements there are, and their motivations, would be valuable information to better understand the phenomenon.

References

Bacon, 2009. *The Art of Community*. O'Reilly, Sebastopol, CA. Available at: http://www.artofcommunityonline.org/downloads/jonobacon-theartofcommunity-1ed.pdf, accessed 23 June 2013.

Bonaccorsi and Rossi, 2006. Comparing motivations of individual programmers and firms to take part in the Open Source movement. From communities to business. *Knowledge, Technology, & Policy*. Winter (2006), Vol. 18, issue 4. pp 40-64.

Ciffolilli, 2004. The economics of open source hijacking and the declining quality of digital information resources: A case for copyleft. *First Monday*, Vol. 9, Number 9, September. Available at: http://firstmonday.org/ojs/index.php/fm/article/view/1173/1093

Dahlander and Magnusson, 2005. Relationships between open source software companies and communities: Observations from Nordic firms. *Research policy* 34 (2005) 481-493.

Dahlander and Magnusson, 2008. How do firms make use of open source communities? *Long range planning* 41 (2008) 629-649.

Daffara, 2011. Open Source License Selection in Relation to Business Models. *Open Source Business Resource*. (February 2011).

Fogel, 2009. *Producing open source software: how to run a successful free software project.* O'Reilly, Sebastopol, CA.

Gamalielsson and Lundell, 2012. Long-Term Sustainability of Open Source Software Communities beyond a Fork: A Case Study of LibreOffice. Hammouda et al. (Eds.): OSS 2012, IFIP AICT 378, pp. 29–47, 2012.

Grad, 2002. A Personal Recollection: IBM's Unbundling of Software and Services, *IEEE Annals of the History of Computing*, Vol. 24, No. 1 (Jan–Mar 2002), pp. 64–71.

Haefliger, von Krogh, and Spaeth, 2008. Code Reuse in Open Source Software. *Management Science*, Vol. 54, No. 1, pp. 180-193.

IBM Archives: 1969, available at: http://www-03.ibm.com/ibm/history/history/year_1969.html, accessed 20 June 2013

Krueger, 1992. Software Reuse. *ACM Computing Surveys*, Vol. 24, No. 2, pp. 131-183.

Lehman, 1980. On Understanding Laws, Evolution, and Conservation in the Large-Program Life Cycle. *Journal of Systems and Software* 1: 213–221.

Lerner and Tirole, 2002. Some Simple Economics of Open Source. *Journal of Industrial Economics*, 52 (June 2002) 197-234.

Levy, 2010. *Hackers: heroes of the computer revolution*. O'Reilly, Sebastopol, CA.

Markus, 2007. The Governance of Free/Open Source Software Projects: Monolithic, Multidimensional, or Configurational?" *Journal of Management and Governance*, 11 (2), 151-163.

Meeker, 2008. *The Open Source Alternative*. Wiley, Hoboken, New Jersey.

Naur and Randell, 1969. Software Engineering, Report on a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th October 1968, Scientific Affairs Division, NATO, Brussels, 138-155. Available at: http://www.cs.dartmouth.edu/~doug/components.txt, accessed 4 March, 2011.

Nyman and Mikkonen, 2011. To Fork or Not to Fork: Fork Motivations in SourceForge Projects. Open Source Systems: Grounding Research, IFIP Advances in Information and Communication Technology Volume 365, 2011, pp 259-268.

Nyman, Mikkonen, Lindman, and Fougère, 2011. Forking: the Invisible Hand of Sustainability in Open Source Software. Proceedings of SOS 2011, available at: http://tutopen.cs.tut.fi/sos11/papers/SOS11_proceedings.pdf#page=9

Nyman, Mikkonen, Lindman, and Fougère, 2012. Perspectives on Code Forking and Sustainability in Open Source Software. *Open Source Systems: Long-Term Sustainability*, 274-279.

Nyman and Lindman, 2013. Code Forking, Governance, and Sustainability in Open Source Software. *Technology Innovation Management Review*, January 2013, pp 7-12.

O'Mahoney, 2007. The governance of open source initiatives: What does it mean to be community managed? *Journal of Management and Governance*, 11 (2), 139-150.

Phipps, 2010. Open core is bad for you. Computerworld, June. Available at http://blogs.computerworlduk.com/simon-says/2010/06/open-core-is-bad-for-you/index.htm, accessed 18 June, 2013.

Raymond, 1999*a*. The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly, Sebastopol, CA. Available online at: http://www.catb.org/esr/writings/homesteading/cathedral-bazaar/

Raymond, 1999*b*. Homesteading the Noosphere. Available at http://catb.org/~esr/writings/homesteading/homesteading/, accessed 20 June 2013.

Raymond, 2001. How To Become A Hacker. Available at: http://www.catb.org/esr/faqs/hacker-howto.html, retrieved 20 June 2013.

Robles and González-Barahona, 2012. A Comprehensive Study of Software Forks: Dates, Reasons and Outcomes. Open Source Systems: Long-Term Sustainability, IFIP Advances in Information and Communication Technology Volume 378, 2012, pp 1-14.

Stallman, 1999. The GNU Operating System and the Free Software Movement. In: DiBona, Ockman and Stone (Eds.), *Open Sources: Voices from the Open Source Revolution*. O'Reilly. Available online at: http://oreilly.com/openbook/opensources/book/, also available at: http://www.gnu.org/gnu/thegnuproject.html, accessed 23 May, 2013

West, 2003. How open is open enough? Melding proprietary and open source platform strategies. *Research Policy* 32 (7), 1259-1285.

West and O'Mahoney, 2005. Contrasting Community Building in Sponsored and Community Founded Open Source Projects," Proceedings of the 38th Annual Hawai'i International Conference on System Sciences, Waikoloa, Hawaii, p. 196c.

West and O'Mahoney, 2008. The Role of Participation Architecture in Growing Sponsored Open Source Communities. *Industries & Innovation*, 15, 2 (April 2008): 145-168.

Wheeler, 2009. F/LOSS is Commercial Software, *Technology Innovation Management Review*, February 2009.

Yin, 2009. *Case Study Research (Fourth Edition)*. Sage, Thousand Oaks, CA.